Universidade Federal de Minas Gerais
Disciplina de Computação Natural
Prof. Gisele L. Pappa

Trabalho Prático 1
Cubo Mágico

Data de Entrega: 12 de setembro de 2016

---

1 Introdução

O cubo de Rubik, também conhecido como cubo mágico, foi inventado no final da década de 70 pelo húngaro Ernõ Rubik e tornou-se um dos ícones da década de 80, época em que foi mais difundido. A versão padrão (Figura 1) consiste de 26 subcubos na parte externa do cubo, cujas partes visíveis são rotuladas com cores. Desses subcubos, seis ficam no centro, 12 na borda e oito nas quinas. O estado do cubo pode ser alterado por meio da rotação de qualquer uma das seis faces. Cada movimento altera a posição de quatro quinas e quatro bordas (pode-se considerar que o centro do cubo sempre permanece na mesma posição). O desafio é alterar o cubo a partir de um estado inicial aleatório até uma configuração em que os todos os rótulos de cada uma das faces sejam da mesma cor.

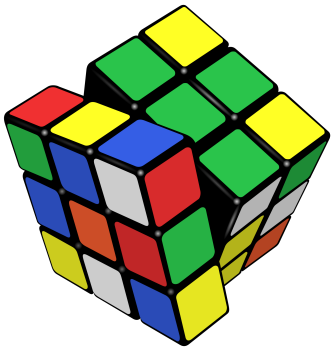

Figura 1: Cubo mágico padrão

Resolver o cubo mágico é uma tarefa difícil. O slogan impresso na caixa original, que diz que existem bilhões de combinações possíveis, é uma considerável atenuação. Na verdade, exitem $4,3 \times 10^{19}$ estados diferentes que podem ser atingidos a partir de qualquer configuração dada.

O objetivo deste trabalho é colocar em prática os conceitos relacionados aos algoritmos evolucionários vistos em sala de aula. Para isso, deve-se implementar um algoritmo genético capaz de resolver algumas instâncias do cubo mágico. A implementação pode ser feita em C, C++, Java ou Python.

Notação: Existem diversas notações para representar a aplicação de movimentos em um cubo mágico. Uma sugestão é utilizar F, B, U, D, L, R para denotar um giro de 90° na

face da frente, do fundo, de cima, de baixo, da esquerda e da direita, respectivamente, em direção horária. Já *Fi*, *Bi*, *Ui*, *Di*, *Li*, *Ri* representariam um giro de 90° nas mesmas faces, mas em direção anti-horária. Giros de 180° seriam denotados por *F2*, *B2*, *U2*, *D2*, *L2*, *R2* e considerados como um movimento único.

A Figura 2 mostra um exemplo da aplicação de três desses possíveis movimentos. A título de curiosidade, provou-se em 2010 que 20 desses movimentos são suficientes para resolver qualquer instância do cubo mágico.

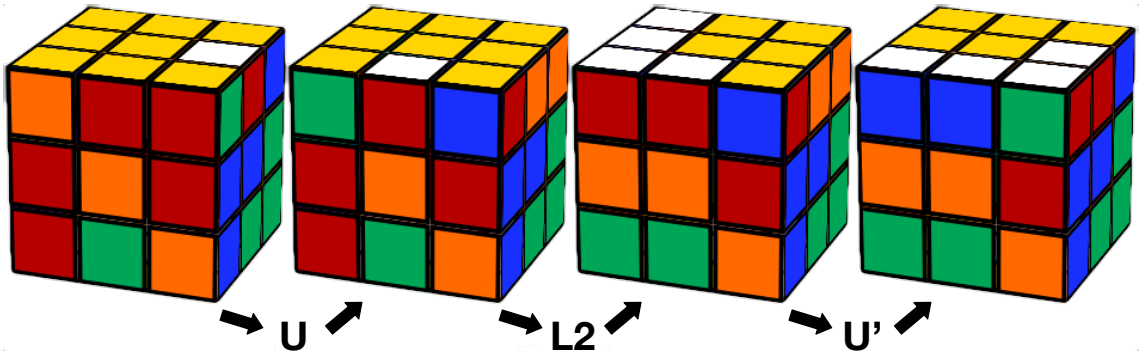

Figura 2: Exemplos de possíveis movimentos

Decisões de Implementação:

1. Como representar um indivíduo (genótipo).
2. Como gerar a população inicial.
3. Como criar uma função de fitness que meça adequadamente o quão próximo da solução um determinado estado está. Uma dica que pode ser útil: uma vez que os subcubos centrais são fixos em relação uns aos outros, a cor objetivo de cada uma das seis faces do cubo é determinada pela cor desses subcubos centrais. Além de criar essa função, deve-se pensar em uma forma de valorizar soluções que minimizam o número de movimentos realizados para alcançar a solução final do cubo.
4. Quais operadores genéticos serão utilizados e de que maneira eles atuarão nos indivíduos. Essa tende a ser uma da partes que exigirão mais atenção, uma vez que a aplicação mal pensada dos operadores pode fazer com que o algoritmo não consiga produzir bons resultados. Além de lidar com a garantia de consistência dos indivíduos, deve-se pensar que o problema de resolver o cubo mágico exige uma lenta, porém constante, exploração do espaço de soluções. Há de se ter em mente, então, o efeito que as operações genéticas terão sobre os indivíduos envolvidos e há de se evitar que tais operações atrapalhem indivíduos promissores.
5. Facilidade de variação de parâmetros – parâmetros *hardcoded* em cantos obscuros do código certamente dificultarão a avaliação a ser feita.

2 Testes

Cinco instâncias de teste serão utilizados neste trabalho, todas disponíveis no Moodle. As três primeiras instâncias tratam de cubos mágicos padrão. A quarta instância representa um cubo mágico 4^3 (conhecido como *Rubik's Revenge*) e a quinta um cubo mágico 5^3 (*Professor's Cube*).

A primeira linha dos arquivos de entrada contém um inteiro indicando o tamanho n de cada uma das dimensões do cubo. As linhas seguintes são divididas em grupos que seguem o seguinte padrão: uma string indicando uma das faces, seguida de n linhas, cada uma contendo n letras. Tem-se então um grupo $n \times n$ de letras representando as cores da face indicada. As cores podem ser codificadas como números, caso isso facilite a criação da função de fitness. O quadro abaixo mostra o arquivo de entrada de uma das instâncias:

```
3
Front
G B B
G O O
G W W
Left
B O R
G G Y
B B R
Right
R O B
W B W
G R Y
Back
O B R
B R R
O W Y
Up
W Y W
Y Y G
Y R Y
Down
W G O
O W Y
O R G
```

3 Metodologia Experimental

A parte de escolha e estudo dos parâmetros deve ser feita da seguinte forma:

- Escolher o tamanho da população e o número de gerações apropriados. O tamanho da população pode ser testado, por exemplo, utilizando 50, 100, 200 e 400 indivíduos.

O número de gerações pode também ser escolhido usando esses mesmos números. Mas como saber se o escolhido é o mais apropriado? Vocês podem avaliar como o aumento no número da população ou de gerações melhora a solução encontrada (em termos do erro gerado), se a população converge, etc.

- Testar duas configurações de parâmetros para crossover e mutação. Na primeira, a probabilidade de crossover (p_c) deve ser alta (por exemplo, 0.9) e a probabilidade de mutação (p_m) deve ser baixa (por exemplo, 0.05). Na segunda, p_c deve ser mais baixa (por exemplo, 0.6) e p_m mais alta (por exemplo, 0.3). Para ambas as configurações, deve-se avaliar o efeito do crossover e da mutação na evolução, isto é, em quantos casos esses operadores contribuem positivamente (os filhos gerados são melhores que os pais) ou negativamente para a evolução? A partir desse estudo inicial, que valores finais você proporia?
- Analisar as mudanças ocorridas quando o tamanho do torneio aumenta de 2 para 5 ou 3 para 7, dependendo do tamanho inicial da população.
- Utilizar elitismo.

Lembrem-se que ao mexer em um dos parâmetros, todos os outros devem ser mantidos constantes, e que a análise dos parâmetros é de certa forma interativa. A configuração de parâmetros raramente vai ser ótima, mas pequenos testes podem melhorar a qualidade das soluções encontradas.

Por ser um método estocástico, a avaliação experimental do algoritmo baseado em PG deve ser realizada com repetições, de forma que os resultados possam ser reportados segundo o valor médio obtido e o respectivo desvio-padrão. A realização de 30 repetições pode ser um bom ponto de partida (lembrando que desvio-padrão alto sugere um maior número de repetições).

Guia para execução dos experimentos

1. Escolha o tamanho da população e o número de gerações. Utilize elitismo, ajuste o tamanho do torneio para 2, p_c para 0.9 e p_m para 0.05.
2. Após alguns testes, mantenha o tamanho da população e o número de gerações e varie p_c e p_m. Os parâmetros escolhidos no passo 1 ainda são apropriados?
3. Definidos o tamanho da população, número de gerações, p_c e p_m, aumente o tamanho do torneio.
4. Escolha os melhores valores para os parâmetros anteriores e retire o elitismo. Os resultados obtidos são os mesmos, melhoraram ou pioraram?
5. Se desejar, teste outras características para resolver o problema.

Estatísticas importantes

Essas estatísticas devem ser coletadas em todas as gerações.

1. Fitness do melhor e pior indivíduos.

2. Fitness média da população.
3. Número de indivíduos repetidos na população.
4. Número de indivíduos gerados por crossover melhores e piores que a fitness média dos pais.

O que deve ser entregue...

- Código fonte do programa (devidamente comentado).
- Documentação do trabalho:
  - Introdução.
  - Implementação: descrição sobre a implementação do programa, incluindo detalhes da representação, fitness e operadores utilizados.
  - Experimentos: Análise do impacto dos parâmetros no resultado obtido pelo AE.
  - Conclusões.
  - Bibliografia.

A entrega DEVE ser feita pelo Moodle na forma de um único arquivo zipado, contendo o código e a documentação do trabalho.

Considerações Finais

- Os parâmetros listados para execução dos experimentos são sugestões iniciais e podem ser modificados à sua conveniência.
- Deve-se dar uma grande importância à analise experimental, pois ela determinará grande parte da nota final do trabalho. Assim sendo, tente reservar um tempo considerável para essa parte. Tente focar na resolução e análise do cubo padrão antes de passar para os cubos maiores.
- Mesmo que o seu programa não seja capaz de resolver sequer um cubo 1^3, tente mostrar que houve dedicação para alcançar o objetivo do problema, apontando o que foi feito e quais foram as dificuldades encontradas. Se possível, explique qual aspecto da implementação atrapalhou o processo de evolução do programa.
- Depois da entrega do trabalho, faremos uma competição em sala de aula para avaliar as diversas decisões de implementação do algoritmo e como a otimização dos parâmetros podem levar ao sucesso ou fracasso dele. Aí, levaremos em consideração se o problema foi resolvido e a quantidade de movimentos necessárias.

Referências

[1] Nain El-Sourani and Markus Borschbach. *"Design and Comparison of two Evolutionary Approaches for Solving the Rubik's Cube"*. Parallel Problem Solving from Nature, 2010.

[2] Michael Herdy and Giannino Patone. *"Evolution Strategy in Action. 10 ES-Demonstra tions"*. International Conference On Evolutionary Computation, 1994.

[3] Nail El-Sourani, Sascha Hauke, and Markus Borschbach. *"An Evolutionary Approach for Solving the Rubik's Cube Incorporating Exact Methods"*. Applications of Evolutionary Computation, 2010.

[4] Todor Balanov. *"Rubik's cube genetic algorithm solver?"*. Asked on Stack Overflow (http://stackoverflow.com/questions/36068550/rubiks-cube-genetic-algorithm-solver).

[5] Ashutosh Tyagi and Poonam Tyagi. *"GEN-R: Genetic Algorithm Based Model for Rubik's Cube Solution Generator"*. International Journal of Science and Advanced Technology, 2011.